Unification of Box Shapes in Molecular Simulations

H. BEKKER

Department of Computing Science, University of Groningen, 9700 AV Groningen, The Netherlands

Received 3 June 1996; accepted 23 June 1997

ABSTRACT: In molecular simulations with periodic boundary conditions the computational box may have five different shapes: triclinic; the hexagonal prism; two types of dodecahedrons; and the truncated octahedron. In this article, we show that every molecular simulation, formulated in one of these boxes, can be transformed into a simulation in one of the other ones. The transformation can be done in a preprocessing phase. The simulation in the new box is exactly identical to the simulation in the original one. This means that every molecular simulation may be done in the same type of box. Because the triclinic box is the easiest one to implement, we pay special attention to how to transform the other four box types into triclinic boxes. As a consequence, simulations in the often used truncated octahedron are superfluous; they may be done in a much simpler way in a triclinic box. © 1997 John Wiley & Sons, Inc. *J Comput Chem* **18**: 1930–1942, 1997

Keywords: molecular simulation; periodic boundary conditions; box shape; lattice

Introduction

To mitigate finite system effects most molecular simulations are done on systems with periodic boundary conditions (PBC). This means that the computational box is surrounded in a space-filling way by replica boxes, with identical content. In terms of the crystallographic Bravais lattices we consider only triclinic systems (i.e., systems without symmetry elements).

Fejes Tóth[1] showed that, in three-dimensional (3-D) space, there are five convex† box types (see Fig. 1) that can be stacked in a space-filling way; that is, there are five possible types of boxes which may serve as a computational box: the triclinic box; the hexagonal prism; two types of dodecahedrons; and the truncated octahedron. For brevity, we will designate these box types PCT1, PCT2, PCT3, PCT4, and PCT5, where PC stands for ''primitive cell,'' and T stands for ''type''. The

†This property is not strictly necessary, but image calculations would become very complex for a nonconvex box.

© 1997 John Wiley & Sons, Inc.
CCC 0192-8651 / 97 / 151930-13

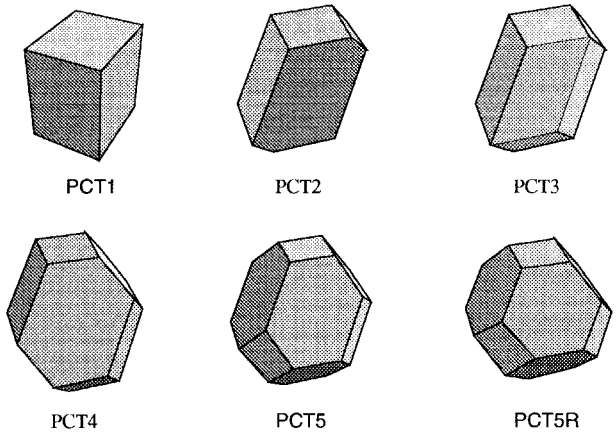

FIGURE 1. Instances of the triclinic box, the hexagonal prism, two types of dodecahedrons, the truncated octahedron, and the most regular instance of the truncated octahedron, in this article designated by PCT1, PCT2, PCT3, PCT4, PCT5, and PCT5R, respectively.

notion ''primitive cell'' will be explained later. The rectangular instance of PCT1 will be designated by PCT1R and the most regular instance of PCT5 by PCT5R. We will use PC to designate, in general, one of the boxes PCT1, PCT1R, PCT2, PCT3, PCT4, PCT5, PCT5R. In the molecular simulation world, PCT5R is often called ''the truncated octahedron,'' but as we will show later it is only the most regular instance of a broader class of boxes.

In the early years of molecular simulation PCT1R was used. Later, PCT3 was introduced,[9] and then PCT5R.[8] Complex-shaped boxes were introduced because it was believed (erroneously) that this was the only way to get a minimal volume simulation. An implicit condition was that the box should contain an *unfragmented* molecule. This superfluous implicit condition has led to the use of complex-shaped boxes. As will become clear from this article, it is not forbidden that the molecule is stored in the box in pieces, provided that the molecule is reconstructed when the boxes are stacked.

In current implementations of molecular simulation algorithms, the shape of the computational box has to be taken into account at many places in the algorithm, notably in neighbor searching, in nonbonded force calculations, in bonded force calculations, and in the part in which particles are reset into the box. For the boxes PCT2 . . . PCT5, which have a complex shape, calculating the position of image particles outside the box as a function of their position in the box is complex. For this reason, in most molecular simulation packages, only a limited set of box shapes has been implemented; for instance, in the molecular dynamics package GROMOS,[2] only limited instances of PCT1 and PCT5R have been implemented.

In this article we will show that every molecular simulation that is formulated in one of the boxes PCT1 . . . PCT5, can be transformed into a simulation in any one of the other boxes. Thus, a simulation formulated in PCT2 . . . PCT5 can be transformed into a simulation in PCT1 or PCT1R. These transformations can be done in a preprocessing stage of a molecular simulation, so the actual simulation can take place in, for example, PCT1 or PCT1R, including neighbor searching, nonbonded force calculations, bonded force calculations, resetting particles into the box, pressure scaling, etc. The simulation in the new box is *exactly* identical to a simulation in the initial, untransformed box. So, for example, the number of particles and interactions to be evaluated is exactly the same in all cases.

It is possible to transform molecular simulations in a simple box into a simulation in a complex-shaped box. However, because for molecular simulations such a transformation is of little interest we will not discuss such transformations.

The structure of this article is as follows. In the next section we define the shape of PCT1 . . . PCT5 in an algebraic way by a lattice and a metric. The lattice-and-metric way of defining PCT1 . . . PCT5 is not suitable for geometrical considerations. Therefore, we introduce a different but equivalent representation of PCT1 . . . PCT5. Using this representation, we show that PCT1 . . . PCT4 are degenerate instances of PCT5. We then show how a tiling of the space with PCT5 defines a lattice. We define a PCT1 and a PCT1R in terms of a given PCT5 such that PCT1 and PCT1R define the same lattice as PCT5. Because PCT1 . . . PCT4 are degenerate instances of PCT5, the same expressions may be used to define a PCT1 and a PCT1R in terms of PCT1 . . . PCT4.[‡] The fourth section shows how to transform a simulation in some box into a simulation in another box. Special attention is paid to transforming particles from one box into the other one. As an example, in the fifth section, we show how a simulation, formulated in PCT5R is transformed into PCT1 and PCT1R.

The fact that every simulation, formulated in some box may be transformed into a simulation in an other box, clarifies a number of unresolved

[‡] Obviously, transforming PCT1 into PCT1 is an identity transformation. But because of the generic character of the algorithms we propose, we do not have to exclude this transformation.

matters—notably, the pressure scaling of simulations in an PCT2 . . . PCT5 box, controlling the long range order of molecular systems, and the maximum allowed cut-off radius. These matters and more are discussed in the last section.

The methods presented in this article may be used to transform existing molecular simulations, formulated in PCT2 . . . PCT5, into a simulation in, for example, PCT1 or PCT1R. That is, however, not the best way to set up a new simulation because then, complex box shapes are still used to set up a simulation. In the subsection ''How to Set Up a Simulation,'' it is shown how to do so without using complex boxes.

We feel that the methods as presented in this article to do molecular simulations in a simple box, together with the efficient method presented elsewhere,[3] will result in faster and simpler molecular simulation software with a wider range of features. All this is brought about, not by improving existing implementations, but by revising the basic concepts of MD simulation.

Defining Boxes

The most natural way to introduce the box types PCT1 . . . PCT5 is by way of a lattice and a metric. In 3D space, a lattice, $\mathscr{L}$, is the set of points:

$$\mathscr{L}(\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}) \equiv n_1\boldsymbol{K} + n_2\boldsymbol{L} + n_3\boldsymbol{M}, \quad \text{with } n_1, n_2, n_3 \in \mathbb{Z} \qquad (1)$$

where $\boldsymbol{K}$, $\boldsymbol{L}$, and $\boldsymbol{M}$ are three independent vectors, called the *basis* vectors. We define a *lattice vector* as a vector connecting two lattice points; therefore, because the origin is a lattice point, lattice vectors are also given by eq. (1). Two points, 1 and 2, are called *corresponding points* when their positions are related by:

$$\boldsymbol{r}_1 + \text{lattice vector} = \boldsymbol{r}_2 \qquad (2)$$

In Euclidean space the squared distance $d^2(p_1, p_2)$ between two points p_1 and p_2 is given by

$$d^2(p_1, p_2) = (\boldsymbol{p}_1 - \boldsymbol{p}_2)^T \mathbf{m} (\boldsymbol{p}_1 - \boldsymbol{p}_2) \qquad (3)$$

with $\mathbf{m}$ a positive definite matrix. Using the metric $\mathbf{m}$, the space can now be partitioned in Voronoi regions or *primitive cells*, where a primitive cell is defined as the set of points closer to some lattice point than to any other lattice point. In this way a tiling of the space is generated with identical tiles. Depending on the value of $\mathbf{m}$ and the lattice vectors $\boldsymbol{K}$, $\boldsymbol{L}$, $\boldsymbol{M}$, the shape of the tiles is one of the box types PCT1 . . . PCT5.[6] In Figure 2, two 2D examples are given of a primitive cell defined by the same lattice but by different distance functions.

It has been shown[4] that a primitive cell, as we defined it, is centrally symmetric, and that it is bounded by pairs of parallel faces. A face is a centrally symmetric hexagon or parallelogram. The edges of a primitive cell consist of groups of parallel lines. In the following we will assume that the primitive cells considered are centered at the origin or are adjacent to an origin-centered primitive cell.

A primitive cell defined as above is an open set of points, because, in our definition of a primitive cell, we do not consider points with the same distance to two lattice points. This would mean that a point with an equal distance to two or more lattice points is not in any primitive cell at all. For molecular simulation this is undesirable; every point in the infinite PBC system should belong to exactly one (image) box. Therefore, it is necessary that we define a primitive cell as a half-open, half-closed set of points, so that tiling the space with primitive cells covers every point of space exactly once. How this is implemented is of no importance in later discussions.

Using a lattice and metric to define a primitive cell is conceptually elegant, but not very well suited for geometrical considerations. Therefore, we will now introduce another way to describe the shape and size of boxes.

We describe PCT5 by giving its edge vectors $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{g}$ (see Fig. 3). These six vectors completely define PCT5, because it consists of 36 edges, which can be grouped into six groups of six parallel edges each. When the vectors $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{g}$ were independent, PCT5 would have $6 \times 3 = 18$ degrees of freedom. However, the vectors defining the hexagonal planes should be coplanar. This gives four conditions: $|\boldsymbol{c}, \boldsymbol{e}, \boldsymbol{g}| = 0$ $|\boldsymbol{b}, \boldsymbol{d}, \boldsymbol{g}| = 0$, $|\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{f}| = 0$; and $|\boldsymbol{b}, \boldsymbol{e}, \boldsymbol{f}| = 0$. So, of PCT5, the shape, size,

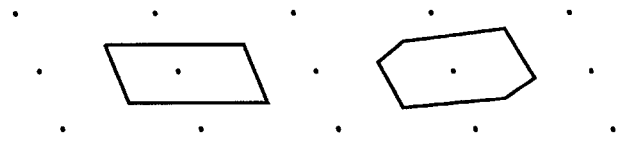

FIGURE 2. Two primitive cells defined by the same lattice but two different distance functions.

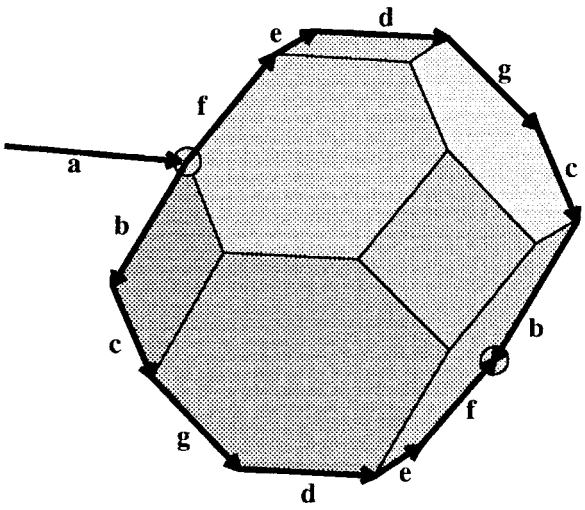

FIGURE 3. An instance of PCT5 defined by the six edges $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{g}$.

and orientation, but not its position, can be described by $18 - 4 = 14$ parameters.

The boxes PCT4 . . . PCT1 can be obtained by degenerating PCT5 as shown in Figure 4. To degenerate PCT5 into PCT4, only the *length* of vector $\boldsymbol{g}$ should go to zero because the direction of $\boldsymbol{g}$ is not free. This is because $\boldsymbol{g}$ is the intersection of the planes defined by the vectors $\boldsymbol{c}, \boldsymbol{e}$ and $\boldsymbol{b}, \boldsymbol{d}$. So, PCT4 has one degree of freedom less than PCT5 (i.e., $14 - 1 = 13$). In the same way, PCT4 can be degenerated into PCT3 by letting $f \to 0$. Again, because $\boldsymbol{f}$ is the intersection of two planes, defined by the vectors $\boldsymbol{c}, \boldsymbol{d}$ and $\boldsymbol{b}, \boldsymbol{e}$, only the *length* of $\boldsymbol{f}$ can be changed. So, PCT3 can be described by $13 - 1 = 12$ parameters. PCT2 can be obtained from PCT3 by choosing the vectors $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}$ to be linearly dependent. This condition brings the number of degrees of freedom of PCT2 to $12 - 1 = 11$. PCT1 can be obtained from PCT2 by $\boldsymbol{e} \to 0$. The vector $\boldsymbol{e}$ is not completely free; it should be in the plane defined by the vectors $\boldsymbol{c}, \boldsymbol{d}$. So it has two degrees of freedom. This brings the number of degrees of freedom of PCT1 to $11 - 2 = 9$, which is what may be expected from a triclinic box.

The whole process of going from PCT5 to PCT1 can be written concisely as:

$$\begin{aligned} \text{PCT5} &\xrightarrow{g \to 0} \text{PCT4} \xrightarrow{f \to 0} \text{PCT3} \\ &\xrightarrow{|cde| \to 0} \text{PCT2} \xrightarrow{e \to 0} \text{PCT1} \end{aligned} \qquad (4)$$

This shows that PCT1 . . . PCT4 are degenerate instances of PCT5, that PCT1 . . . PCT3 are degenerate instances of PCT4, etc. So, PCT5 is the generic space filler. Therefore, in the following paragraphs we only consider transformations of PCT5 Some properties of PCT5 . . . PCT1 are given in Table I.

Again, something about notation: Until now we only have been speaking about different box *types*. In this and the following sections different ways to *describe* boxes are introduced. We will denote a box described by the vectors $\boldsymbol{b, c, d, e, f, g}$ as PCDg. The D stands for ''described,'' and g stands for ''general,'' because this is the most general way to describe a primitive cell. The box PCDg can be of any type. Later, two other ways to describe boxes will be introduced.

Constructing Simple Boxes

In this section, we will propose two boxes, a triclinic and a rectangular one, generating the same

TABLE I.
Some Properties of PCT5 . . . PCT1.

	PCT5	PCT4	PCT3	PCT2	PCT1
No. faces	14	12	12	8	6
No. rhombi	6	8	12	6	6
No. hexagons	8	4	0	2	0
No. edges	36	28	24	18	12
No. vertices	24	18	14	12	8
Degrees of freedom	14	13	12	11	9

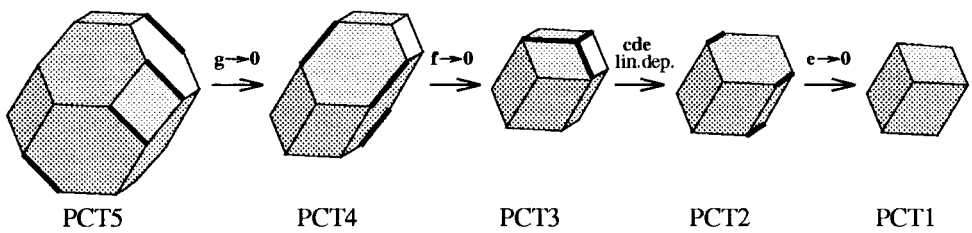

FIGURE 4. PCT5, PCT4 created by letting $\boldsymbol{g} \to 0$ of PCT5, PCT3 created by letting $\boldsymbol{f} \to 0$ of PCT4, PCT2 created by letting $|\boldsymbol{cde}| \to 0$ of PCT3, PCT1 created by letting $\boldsymbol{e} \to 0$ of PCT2. Thick lines go to zero.

lattice as an initial box PCDg. We will first derive expressions for the lattice vectors of the lattice generated by a box PCDg.

We assume that a box PCDg is centered at the origin and that three translated copies of it are fitted along whole faces to it. The centers of these three copies are at:

$$\boldsymbol{K} = (\boldsymbol{g} + \boldsymbol{d} + \boldsymbol{e} + \boldsymbol{f}), \quad \boldsymbol{L} = (\boldsymbol{g} + \boldsymbol{b} + \boldsymbol{e}), \quad \boldsymbol{M} = (\boldsymbol{f} - \boldsymbol{c} + \boldsymbol{e}) \tag{5}$$

The original box and two surrounding copies are shown in Figure 5.

With some patience, it can be verified that every other replica box fitted to the original box is shifted over an integer linear combination of $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$. *So, the whole space can be tiled with copies of the original box centered at the lattice points defined by* $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$. As long as the vectors $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ are linearly independent the expressions [eq. (5)] for $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ are meaningful; that is, they also hold for the boxes PCT1 ... PCT4.

With the lattice vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ we can define a primitive cell which generates the lattice defined by $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$, namely the triclinic box spanned by the lattice vectors themselves. We will call a box defined by the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$, ''PCDKLM.'' The box PCDKLM can only be of the type PCT1.

Now we will introduce a rectangular box that generates the lattice defined by the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$. First, the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ have to be reordered such that:

$$|\boldsymbol{K}| \geq |\boldsymbol{L}| \geq |\boldsymbol{M}| \tag{6}$$

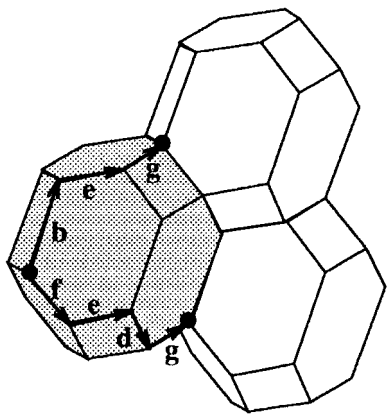

FIGURE 5. The vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ defined by PCT5 with three replica boxes fitted along whole faces. It can be seen by inspection that $\boldsymbol{K} = (\boldsymbol{g} + \boldsymbol{d} + \boldsymbol{e} + \boldsymbol{f})$, $\boldsymbol{L} = (\boldsymbol{g} + \boldsymbol{b} + \boldsymbol{e})$, $\boldsymbol{M} = (\boldsymbol{f} - \boldsymbol{c} + \boldsymbol{e})$ (not visible).

As we will show in Appendix B this simplifies some calculations in a later stage. Using the reordered vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$, the vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ spanning a rectangular primitive cell are given by a Gram–Schmidt orthogonalization process (see Fig. 6):

$$\boldsymbol{U} = \boldsymbol{K}, \quad \boldsymbol{V} = \boldsymbol{L} - (\boldsymbol{L} \cdot \hat{\boldsymbol{K}})\hat{\boldsymbol{K}}, \quad \boldsymbol{W} = (\boldsymbol{M} \cdot \boldsymbol{K} \hat{\times} \boldsymbol{L})\boldsymbol{K} \hat{\times} \boldsymbol{L} \tag{7}$$

with $\hat{\boldsymbol{a}} \equiv \boldsymbol{a}/a$ and with $\boldsymbol{b} \hat{\times} \boldsymbol{c} \equiv \boldsymbol{b} \times \boldsymbol{c}/|\boldsymbol{b} \times \boldsymbol{c}|$. The second expression means that $\boldsymbol{V}$ is perpendicular to $\boldsymbol{K}$, also to $\boldsymbol{U}$, and that it is in the plane defined by $\boldsymbol{K}$ and $\boldsymbol{L}$. The third expression means that $\boldsymbol{W}$ is perpendicular to the plane defined by $\boldsymbol{K}$ and $\boldsymbol{L}$, which implies that it is perpendicular to the plane defined by $\boldsymbol{U}$ and $\boldsymbol{V}$. Analogously with the nomenclature already introduced, we will call the primitive cell described by the vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ PCDUVW. The box PCDUVW can only be of the type PCT1R.

Boxes should be centered at lattice points, and thus, should be stacked *with relative shifts over the lattice vectors* $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$. This means that, in a tiling with the boxes PCT1R, the boxes are not fitted along whole faces (see Fig. 7). This last fact may look a bit strange because, with the primitive cells PCT1 ... PCT5, the space could be tiled by fitting these boxes along whole faces. A way out of this seemingly strange property of PCT1R is by taking it as a PCT5, with some of its faces in the same plane. In general PCT5 has contact along whole faces with 14 adjacent boxes, just like PCDUVW. So, by taking PCT1R as a special instance of PCT5 the anomaly is explained.

Let us now briefly look at the volume of the primitive cells. For a given lattice, defined by the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$, the volume of the primitive cells

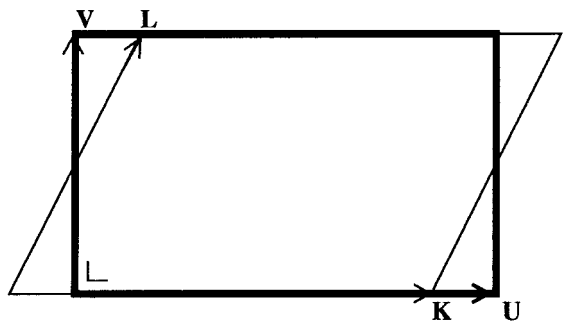

FIGURE 6. A rectangular primitive cell PCDUVW (thick lines), and the PCDKLM from which it is derived (thin lines), both centered around the same point.

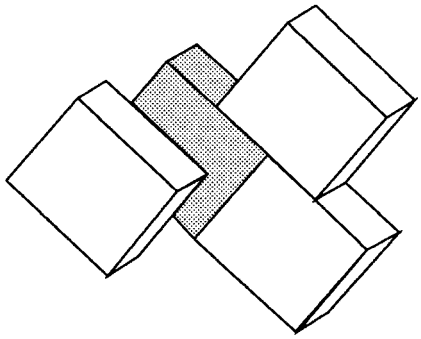

FIGURE 7. The box PCT1R has to be centered at lattice points, resulting in a tiling that is seemingly not a tiling along whole faces. However, by taking PCT1R as a special instance of PCT5, this tiling may be taken as a face-to-face tiling. This is indicated by the fact that every box PCT1R is directly surrounded by 14 boxes, just like a tiling with a general PCT5.

PCDg, PCDKLM, and PCDUVW is the same, and is given by determinant $|\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}|$. That is because to every lattice point belongs one primitive cell, no matter the shape of this primitive cell. So, in terms of lattice and metric, the volume of a primitive cell does not depend on the metric.

Transforming Molecular Simulations

In the previous section we showed how any PC from PCT1 . . . PCT5 can be transformed into a PCT1 and PCT1R defining the same lattice as PC. We will now look at these boxes as molecular simulation boxes; that is, as boxes filled with particles, and show how to transform such a box PC with particles into another box PC′ with particles, such that there is no difference between the molecular simulation of these two systems.

Let us first define when two molecular systems are identical. Therefore, we consider two boxes, PC and PC′, where both boxes contain the same set of particles. With IS and IS′ we mean the two infinite molecular systems, generated by tiling the space with PC and PC′, respectively, including the particles they contain. The molecular simulations in PC and PC′ are called *identical* when the particles in IS and IS′ coincide.

We can now formulate a relation between PC and PC′, and a relation between the position of the particles in PC and PC′, such that the simulations in PC and PC′ are identical. These relations are the cornerstones of this article.

Theorem

Two molecular simulations, formulated in two boxes, PC *and* PC′, *are identical when*:

- PC *and* PC′ *define the same lattice; and*
- *the particles in* PC *and* PC′ *are at corresponding positions.*

Proof: Assume that IS is generated by tiling the space with PC with particles. Let us look at a particle i in PC. $\mathbf{r}_i$ may be located outside PC′, but there is a unique shift over a lattice vector translating $\mathbf{r}_i$ into PC′. The position of i in PC′ will be called $\mathbf{r}'_i$. (The shift is unique because a primitive cell does not contain corresponding points.)

We now tile the space with PC′, containing the possibly shifted particle i. Because $\mathbf{r}_i$ and $\mathbf{r}'_i$ only differ by a lattice vector, and tiling the space implies shifts over all lattice vectors, the set of points generated by all lattice shifts of $\mathbf{r}_i$ and the set of points generated by all lattice shifts of $\mathbf{r}'_i$ coincide. This proof also holds when going from PC′ to PC.

In the previous sections we saw how to transform the boxes PC and PC′ into each other. The remaining problem is how to transform particles from the initial box PC into the box PC′. According to the second part of the theorem, particles should be shifted over lattice vectors. In general, it is difficult to give an explicit expression for the required shift. Therefore, we will not use a direct method to find the required shift over a lattice vector, but rather try lattice vectors. This can be done because it is possible to give an upper bound of the order of the required shift; that is, if the required shift is $n_1\boldsymbol{K} + n_2\boldsymbol{L} + n_3\boldsymbol{M}$, it is possible to give an upper bound of n_1, n_2, n_3. In Appendix A it is proved that particles in PCDg have to undergo, at most, first order shifts to be translated into PCDKLM (i.e., $-1 \leq n_1, n_2, n_3 \leq 1$). This way of determining the required lattice vectors is not the most efficient, but it is general. Because the process of translating particles from PC into PC′ is done in a preprocessing stage of the actual molecular simulation, the inefficiency is no problem.

Algorithm for Translating Particles

We will now discuss an algorithm to move particles from PCDg into the related PCDKLM. (The algorithm to translate particles into PCDUVW is analogous.)

A naive implementation would consist of a triple loop, generating all linear combinations of the lattice vectors $\boldsymbol{K}$, $\boldsymbol{L}$, and $\boldsymbol{M}$. However, such an implementation would be inefficient, because the most improbable shifts, being shifts over long lattice vectors, are tried first, whereas the most probable shift, being no shift, is tried last. Later, we will encounter a case where the maximum shift is more than one, which results in even more inefficiency. Therefore, we give an algorithm which starts with trying zeroth order shifts. Then, the second most probable shifts are tried, which are shifts over lattice vectors in the first layer around the origin. Then, if *max_order* > 1, the third most probable shifts are tried, and so on.

We assume that we have a boolean function `INPCDKLM(r)`, which determines whether `r` is in the box PCDKLM. With this function, and using the boundedness of the required translations, the algorithm to move a particle from PCDg into PCDKLM is as follows:

```
procedure PutIntoPCDKLM(var r: vector);
  var
    j, a, b, maxRadius: integer;
  procedure tryShiftingIntoBox(n1, n2, n3: integer);
    var
      shift: vector;
    begin      {note vector operations}
      shift := n1*k + n2*l + n3*m;
      if InPCDKLM(r + shift) then begin
        r := r + shift;
        exit(PutIntoPCDKLM);
      end; {if}
      if InPCDKLM(r-shift) then begin
        r := r-shift;
        exit(PutIntoPCDKLM);
      end; {if}
  end; {tryShiftingIntoBox}

begin{PutIntoPCDKLM}
  maxRadius := 100;
  for j := 0 to maxRadius do begin {try further
  and further away}
    for a := -j to j do begin
      for b := -j to j do begin
        tryShiftingIntoBox(a, b, j);
      end; {for b}
    end; {for a}
    for a := -j to j do begin
      for b := -j + 1 to j - 1 do begin
        tryShiftingIntoBox(a, j, b);
      end; {for b}
    end; {for a}
    for a := -j + 1 to j - 1 do begin
      for b := -j + 1 to j - 1 do begin
        tryShiftingIntoBox(j, b, a);
      end; {for b}
    end; {for a}
  end; {for j}
  FatalError('Max radius overflow.');
end; {PutIntoPCDKLM}
```

Comments on this pseudo code: Note that we use vector operators in this code. The code consists of three similar blocks, each consisting of a nested loop over `a` and `b`. In the first block, all lattice points in the top and bottom plane of a cube with ''radius'' `j` are visited. In the second block, the lattice points in the left and right plane are visited. In the third block, the lattice points in the front and back plane are visited.

In Appendix B it is shown that particles in PCDKLM have to undergo, at most, second order shifts to be translated into PCDUVW. This means that the procedure proposed in this subsection can also be used for that case, with of course the exception that, in the algorithms, `maxOrder := 2`. Later we will encounter a case where the maximum order of the translation is unbounded, but still zero shifts are the most probable ones with decreasing probability outward.

Example Transformation of a Simulation

In the MD simulation package GROMOS, two box shapes are implemented: PCT1R and PCT5R. In this section, as an example application of the theory, we will show how a simulation, formulated in PCT5R, can be transformed into a simulation in PCT1 and PCT1R.

PCT5R is obtained by cutting away pieces of a cube with edge lengths h. This results in a PCT5R with edge vectors $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$ given by:

$$\boldsymbol{b} = \begin{pmatrix} 0 \\ -\frac{1}{4}h \\ -\frac{1}{4}h \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} 0 \\ \frac{1}{4}h \\ -\frac{1}{4}h \end{pmatrix}, \quad \boldsymbol{d} = \begin{pmatrix} -\frac{1}{4}h \\ \frac{1}{4}h \\ 0 \end{pmatrix} \quad (8)$$

$$\boldsymbol{e} = \begin{pmatrix} -\frac{1}{4}h \\ -\frac{1}{4}h \\ 0 \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} -\frac{1}{4}h \\ 0 \\ \frac{1}{4}h \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} -\frac{1}{4}h \\ 0 \\ -\frac{1}{4}h \end{pmatrix} \quad (9)$$

Applying (5) gives the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$:

$$\boldsymbol{K} = \begin{pmatrix} -h \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{L} = \begin{pmatrix} -\frac{1}{2}h \\ -\frac{1}{2}h \\ -\frac{1}{2}h \end{pmatrix}, \quad \boldsymbol{M} = \begin{pmatrix} -\frac{1}{2}h \\ -\frac{1}{2}h \\ \frac{1}{2}h \end{pmatrix} \quad (10)$$

Applying (7) gives the vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$

$$\boldsymbol{U} = \begin{pmatrix} -h \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{V} = \begin{pmatrix} 0 \\ -\frac{1}{2}h \\ -\frac{1}{2}h \end{pmatrix}, \quad \boldsymbol{W} = \begin{pmatrix} 0 \\ -\frac{1}{2}h \\ \frac{1}{2}h \end{pmatrix} \quad (11)$$

It can be checked that the volume of each of these three figures (PCT5R, PCT1, PCT1R) is $\frac{1}{2}h^3$.

In Figure 8a, PCT5R is shown with a (fancy) spherical molecule. The molecule is mapped into PCT1 according to the theorem in the previous section (see Fig. 8b). The fact that the molecule is ''cut into pieces'' in PCT1 indicates that the atoms of the molecule are shifted over different lattice vectors when translated from PCT5R into PCT1.

PCT5R is the most regular instance of PCT5. Consequently, as can be seen in eq. (10), the lattice vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ are also special; that is, to create image particles surrounding the original box PCT5R the particles in the box have to undergo regular shifts. The regularity of these shifts is exploited in ref. 5 to calculate, in a simple way, the required shifts. Quite appropriately, this shift pattern is called the ''checkerboard'' periodic boundary condition. However, this shift method is only applicable to PCT5R, and the actual simulation is still done in PCT5R.

We have made some software available[§] as both Turbo Pascal and C code with executables. In DEM1, the primitive cells PCT1 ... PCT5 can be (randomly) generated, and visualized (in X). In DEM2 the process of moving particles from PCT5R into PCDKLM and PCDUVW is implemented. DEM2 can thus be used by the MD community to transform existing simulations, formulated in PCT5R, into a simulation in PCDKLM and PCDUVW.

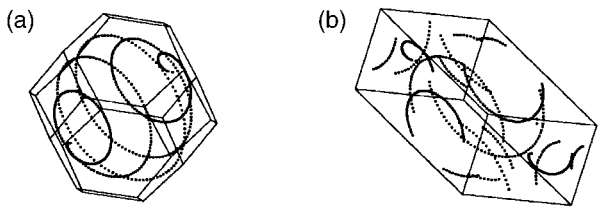

FIGURE 8. (a) PCT5R with a (fancy) spherical molecule. (b) PCT1 derived from PCT5R, with the molecule mapped into it. It is instructive to copy (b) on a transparent sheet, and to fit this copy at various faces to its original. It can then be seen that the molecule is reconstructed.

[§] Can be obtained by anonymous ftp from ftp.cs.rug.nl in the directory pub / mdbox.

Related Topics

PRESSURE SCALING

The pressure of a molecular system can be represented by a 3×3 tensor $\mathbf{P}$. The scalar pressure P is defined as:

$$P \equiv \frac{1}{3}\text{trace}(\mathbf{P}). \quad (12)$$

In many MD simulations, every now and then the MD system—that is, the box and particle positions—is scaled depending on the most recently calculated pressure. In case the computational box is triclinic, it is well known how to scale the system[10]: in case only the scalar pressure is calculated, the box and particles are scaled in every dimension with the same factor. In case the pressure is calculated per dimension, the system is scaled per dimension, proportional to the components of the pressure vector. In case the full tensorial pressure is used, the system is scaled by multiplying all particle position and box vectors with the scaled pressure tensor. As a result of the last two types of pressure scaling, the angles of the system may change.

With the notions developed in this article, it is clear how to scale the system when the computational box is one of PCT2 ... PCT5 and pressure scaling per dimension or a full tensorial pressure is used. Then, just as in the case of a triclinic box, *the system may be scaled by scaling box vectors and particle positions per dimension by multiplying box vectors* $(\boldsymbol{b} \ldots \boldsymbol{g})$ *and particle positions with a scaled tensor* $\mathbf{P}$, respectively. This is because relations (5) and (7) are linear.

LATTICE REDUCTION

Until now our attention has been focused on transforming simulations in a complex box into simulations in a simple box; that is, on transformations between different box types. We will now discuss a transformation from one PCT1 into another PCT1, both defining the same lattice.

Let us suppose that a 2D simulation of a long thin molecule is set up as shown in Figure 9a. In

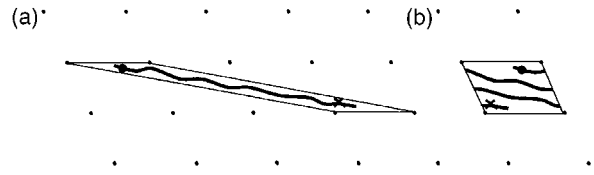

FIGURE 9. A 2-D example of two primitive cells of the same type (parallelogram), defining the same lattice. Applying lattice reduction to (a) gives (b), resulting in a cell with shorter spanning vectors than the original primitive cell. The molecule in (a) is mapped into (b) according to Theorem 2.

principle, the simulation may be done in this box, but for a number of practical reasons this may be unattractive; for example, the cut-off sphere may then be located in many boxes at the same time. To improve this situation, a general technique, called lattice reduction,[6] may be applied.

According to the theorem given earlier, a simulation may be done in every box that defines the same lattice as the original box. When we assume that the original box defines the lattice basis vectors $\boldsymbol{K}$, $\boldsymbol{L}$, the same lattice is defined by the basis vectors $\boldsymbol{K}$, $\boldsymbol{L} - n\boldsymbol{K}$ with $r \in \mathbb{Z}$. So, *the simulation may just as well be done in a box defined by the vectors* $\boldsymbol{K}$, $\boldsymbol{L} - n\boldsymbol{K}$. When the particles are moved from the original box to the new one this results in the system as shown in Figure 9b. This method may be generalized to 3D.

Let us now be a bit more precise. For a given box, PCT1, spanned by the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$, we look for three vectors $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$, such that the vectors $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$ define the same lattice as the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$. Moreover, the vectors $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$ should span a ''nice'' box, where nice means something like ''as cubic as possible.'' The process of transforming the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ into the vectors $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$ is called lattice reduction. Many different notions of ''reduced'' exist in the literature, but, roughly speaking, they all mean that the cell $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$ is as cubic as possible. It has been shown[6] that, in 3D, the three shortest, linearly independent lattice vectors are a basis of the lattice. Therefore, we will define a reduced basis as: *a reduced basis consists of the three shortest, linearly independent lattice vectors*.

After the process of lattice reduction, particles from the box $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ should be mapped into the box $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$. This should be done according to the theorem; that is, particles should be shifted over lattice vectors. Which lattice basis is used, $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ or $\boldsymbol{K}', \boldsymbol{L}', \boldsymbol{M}'$, does not matter because both are a basis of the same lattice. The algorithm given earlier may be used to shift particles over the required lattice vectors, although, unlike the situation in this section, now there is no upper limit on the required shift (called `max_shift` in the algorithm).

Lattice reduction may be used to answer the question: *for a given triclinic box spanned by the (unreduced) vectors* $\boldsymbol{K}$, $\boldsymbol{L}$, $\boldsymbol{M}$, *how large may the cut-off radius be at most, such that no particle has interactions with two corresponding particles*? This may be reformulated as: how large may the cut-off sphere be at most, such that it does not contain corresponding points? As can be seen in Figure 10a, it is not enough that $max\,\boldsymbol{R}_{co} = \frac{1}{2} min\,(|\boldsymbol{K}|, |\boldsymbol{L}|, |\boldsymbol{M}|)$. Using our foregoing definition of ''reduced basis,'' the answer is:

$$max\,\boldsymbol{R}_{co} = \tfrac{1}{2} min\,(|\boldsymbol{K}'|, |\boldsymbol{L}'|, |\boldsymbol{M}'|) \qquad (13)$$

that is, the cut-off radius should be less than half the length of the shortest *reduced* lattice basis vector (Figure 10b).

LONG-RANGE ORDER

Stacking boxes in a space-filling way introduces a well-defined long-range order in the infinite system. This long-range order may influence the results of a simulation. For example, when the box shape is chosen such that it defines a long-range order close to the long-range order of ice, it may happen that, in a simulation of pure water, the water freezes above 0°C. By simulating water in a box with a long-range order incompatible with the long-range order of ice, the water may be liquid below 0°C. Probably, for every solvent, and depending on the type of simulation, there is an optimal long-range order, so that the solvent behaves normally. So, when setting up a simulation, the resulting lattice must be compatible with the

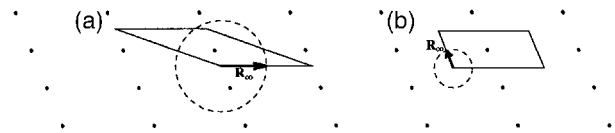

FIGURE 10. (a) A 2-D example of an unreduced primitive cell. When R_{co} is chosen as half the length of the shortest vector spanning the primitive cell, the cut-off sphere still contains corresponding particles. (b) When R_{co} is chosen as half the length of the shortest vector spanning the *reduced* primitive cell, the cut-off sphere does not contain corresponding particles.

desired long-range order. This means that the shape of the computational box is not completely free any longer.

HOW TO SET UP A SIMULATION

From the foregoing it will be clear that a molecular simulation can be done without using complex boxes. We will now show that setting up a simulation can also be done without using complex boxes; that is, we will show that it is not necessary to set up a simulation in a complex box which is subsequently transformed into a simulation in a simple box.

Let us assume that one single large molecule has to be simulated in a solvent. The molecule has been given, and the solvent has to be added when a box has been constructed around the molecule. We will designate this molecule by ''mol'' (Fig. 11). In general, a molecule is not allowed to interact with its own image molecules, so, in the infinite system, the smallest distance between two atoms of two different images of mol should be at least R_{co} apart. For this purpose we surround mol by an enlarged convex hull, such that no atom of mol is closer than $1/2\,R_{co}$ to this enlarged hull. We will designate this enlarged hull of mol by MOL. Three replicas of MOL, with the same orientation as MOL, are designated by MOL′, MOL″, and MOL‴.

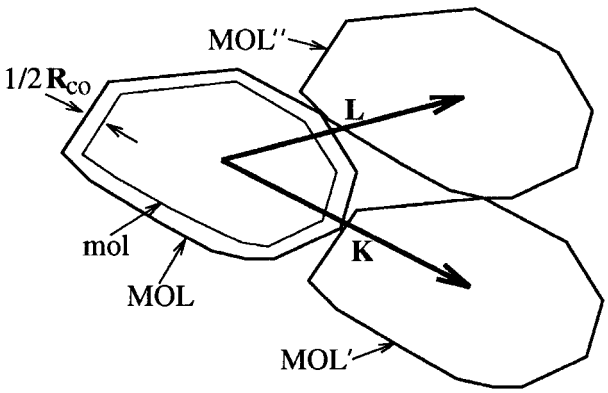

FIGURE 11. To find a computational box with a minimal volume, containing a single molecule, MOL, three translates of MOL have to be fitted to MOL, defining three vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$, such that the volume of box defined by $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ is minimal. After finding such a minimal box, the atoms of MOL can be translated into this box by shifts over lattice vectors, where the lattice is defined by $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$.

To set up a PBC simulation with a minimal amount of solvent means that we have to find the densest lattice packing of translations of MOL. A practical approach to this minimization problem is to fit MOL′ . . . MOL‴ to MOL, such that the volume of the tetrahedron defined by these four molecules is minimal. More exactly, when we define the vector $\boldsymbol{K}$ as the vector connecting the center of MOL with the center of MOL′, the vector $\boldsymbol{L}$ as the vector connecting MOL with MOL″, and the vector $\boldsymbol{M}$ as the vector connecting MOL with MOL‴, the problem boils down to: minimize $|\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}|$ subject to the condition that each of the molecules MOL . . . MOL‴ is touched[‖] by the other three. This is a minimization problem in three parameters. It can be seen as follows: because MOL′ has to touch MOL, the position of MOL′ is determined by two angles, say θ and ϕ, where the origin of these two angles is somewhere in MOL. MOL″ should touch MOL and MOL′, so there is only one degree of freedom in the placement of MOL″. Finally, MOL‴ has to touch the first three, so the placement of MOL‴ is completely determined by the positions of MOL . . . MOL″.

Thus, we have a minimization problem in three variables (minimize $|\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}|$) subject to six contact conditions (contact between every pair of MOL . . . MOL‴). A near minimal solution can be found by a standard minimization procedure such as, for example, NAG routine `E04UCF`. When a minimal volume configuration of MOL . . . MOL‴ has been found, the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ are the vectors defining the triclinic simulation box.[¶] By shifts over lattice vectors, the atoms of mol can now be brought into this triclinic box, and the empty space can be filled with solvent. Of course, if desired, this box can be transformed into a rectangular box as described earlier in this article.

WHICH BOX TO USE: TRICLINIC OR RECTANGULAR?

The main message of this article is that complex-shaped boxes with particles, as, for example, PCT5 and its degenerates PCT4 . . . PCT2, can be transformed into simpler ones; that is, into PCDKLM and PCDUVW. Which one of these last two is the best one as a simulation box is not very clear.

[‖] It is a well-known property of the densest lattice packing of convex figures that every figure is touched by 12 others.

[¶] Obviously, when the simulation has to be set up with a predefined long-range order the optimization process may be skipped.

The choice may be influenced slightly by some parts of the simulated system and the simulation methods used. We will briefly discuss some of these aspects. Still, this discussion will not lead to a strong preference.

Neighbor searching as has been shown in Ref. 7, using a grid search technique significantly improves the efficiency of neighbor searching. The essence of the grid search technique is that a grid is constructed in the computational box, and that for every particle it is determined in which grid cell it is located. Neighbor searching for a given particle then boils down to inspecting its own and directly neighboring grid cells for neighboring particles. In Ref. 7 it was shown that a grid size of $L = \frac{1}{2}R_{co}$ gives an optimal neighbor searching speed, which is six times faster than neighbor searching without using a grid. However, as far as we can now see, the grid search technique can only work efficiently when the grid cells are rectangular or, even better, cubic. Obviously, a rectangular box can be partitioned in cells in a natural way. This does not hold for the nonrectangular box, PCDKLM, because then many grid cells will be empty. Therefore, we think that, in case neighbor searching is implemented with the grid search technique, the rectangular box is to be preferred over the triclinic box. Of course, although the box is rectangular, image particles are created by shifting particles over lattice vectors, and not over the orthogonal vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$.

The function `inbox(r)`. Every now and then during an MD simulation, particles that moved outside the computational box have to be reset into the box. To check whether a particle is inside or outside the computational box, the boolean function, `inbox(r)`, is used. When $\mathbf{r}$ is inside the box the function returns true. Obviously, when PCT1R is used as a computational box, and the directions of $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ coincide with the x, y, z axes, `inbox(r)` can be implemented by checking independently in three directions in what range the components of $\mathbf{r}$ are. This does not work that easily in case of a nonrectangular triclinic box. In that case, a linear transformation on $\mathbf{r}$ has to be done, or some other more complex calculation. So, for the implementation of `inbox(r)` it is desirable to work with PCDUVW.

Full pressure scaling. As we explained before, three kinds of pressure scaling are possible in an MD simulation: uniform in every direction; scaling per dimension; and by using the full pressure tensor. In general, the last two ways of pressure scaling will change the directions of the vectors spanning the computational box. This means that, when PCDUVW is used as a computational box, the angles between the vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ will change, that is, afterwards, the box will not be rectangular any longer. Then, in principle, the function `inbox(r)` will not work properly, and the search grid will no longer be rectangular. However, the computational effort of recalculating the vectors $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ and resetting particles is small, so possibly pressure scaling will not be an obstacle for using a rectangular box.

Summarizing one may say that a rectangular box simplifies the implementation of some parts of molecular algorithms [grid search, `inbox(r)`], but causes small complications in the implementation of other parts.

Conclusion

In this article we studied the possible shapes of the computational box of molecular simulations with PBC. For this purpose, five types of boxes are suitable: triclinic; the hexagonal prism; two types of dodecahedrons; and the truncated octahedron (for short PCT1 . . . PCT5). We showed that PCT1 . . . PCT4 are degenerate instances of PCT5.

The main purpose of this article is to show that, for every simulation in some type of box, simulations in the other four types can be devised that give exactly the same simulation result; that is, it is shown that a simulation in a box with a complex shape may be done in a box with a simple shape. Therefore, we first showed how to transform the complex-shaped box into a triclinic one, and how to transform the triclinic one into a rectangular one. Then we showed how to map particles from the complex-shaped box into the simpler ones.

Important conceptual tools in this article are lattices and primitive cells. It was shown that a simulation box may be taken as a primitive cell. Tiling the space with a box with particles gives an infinite molecular system. In the theorem presented in the fourth section, conditions are formulated on the transformations of boxes and particles.

Although most of this article is about transforming simulations in boxes with a complex shape with particles into simple-shaped boxes, this does not mean that a molecular simulation should be set up in a complex box which is subsequently transformed into a simpler box. On the contrary, because every simulation in a complex box can be transformed in a simpler one in a triclinic box,

nothing is lost when a simulation is set up right away in a triclinic box. In the subsection, ''How to Set Up a Simulation,'' it is explained how this can be done.

With the concepts developed in this article, some matters are clarified. These include, among others, pressure scaling in a complex box and the long-range order introduced by the shape of the box.

Acknowledgments

This research was initiated by the insights of Berend Reitsma. Discussions with Wim Hesselink led to the use of an alternative metric. The discussions with Michael Renardus about the text and the figures improved this article significantly. The comments of a referee made this article more readable.

Appendix A

We will prove that particles in the box PCDg have to be shifted, at most, over first order shifts; that is, over $n_1\boldsymbol{K} + n_2\boldsymbol{L} + n_3\boldsymbol{M}$, with $-1 \leq n_1, n_2, n_2 \leq 1$, to be located in the related PCDKLM. It is instructive to see that this does not hold in general (see Fig. 12); that is, particles in a primitive cell, PC, potentially require infinite shifts to be located in a related PC', where ''related'' means that the cells define the same lattice.

The reason that between PCDg and PCDKLM at most first order shifts are required, has to do with the special choice of $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$. Consequently, when PCDg is long and thin, PCDKLM is also long and thin and is oriented in the same direction. In this way, PCDg and PCDKLM have a large overlap, so that, in going from one to the other cell, only limited particle shifts are required.

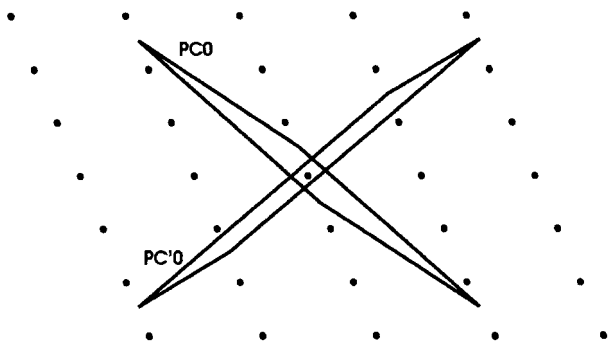

FIGURE 12. Particles in PC have to be shifted over more than first order shifts to be translated into PC'.

Let us now give a more formal proof. Using the 3D nomenclature, we give the proof for the 2D case, but it is simple to extend it to 3D. We start with PCDg and its eight first order images. We will call this arrangement of nine tiles ''F'' (Fig. 13). In this same figure the related PCDKLM is drawn. It is constructed by scaling the rhombus A,B,C,D with a factor 1/2. The rhombus defined by A,B,C,D is in F because the boundary of the rhombus A,B,C,D is in F. This last fact is because, when two space fillers (either 2D or 3D) are fitted face to face, the line connecting their centers of symmetry lies in these two figures. Because PCDKLM is in the rhombus A,B,C,D, it is also in F. The particles in F are shifted from PCDg over at most first order lattice vectors; therefore, the particles in PCDg have to be shifted over, at most, first order shifts to be located in PCDKLM.

Appendix B

In this appendix we will show the necessity of ordering the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ before calculating $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$. We first show what may go wrong when this is not done. Just as in Appendix A, using the 3D nomenclature, we will do this for 2D.

Suppose that we have a PCDKLM as shown in Figure 14. We can construct a rectangular primitive cell from PCDKLM in two ways: by $\boldsymbol{U} = \boldsymbol{K}$ and $\boldsymbol{V} \perp \boldsymbol{U}$ (Fig. 14a); and by $\boldsymbol{U} = \boldsymbol{L}$ and $\boldsymbol{V} \perp \boldsymbol{U}$. (Fig. 14b). From these figures it is clear that PCDUVW has a large overlap with PCDKLM when the longest one of the pair $\boldsymbol{K}, \boldsymbol{L}$ is defined as $\boldsymbol{U}$. So, ordering $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ prevents possibly infinite shifts of particles going from PCDKLM into PCDUVW.

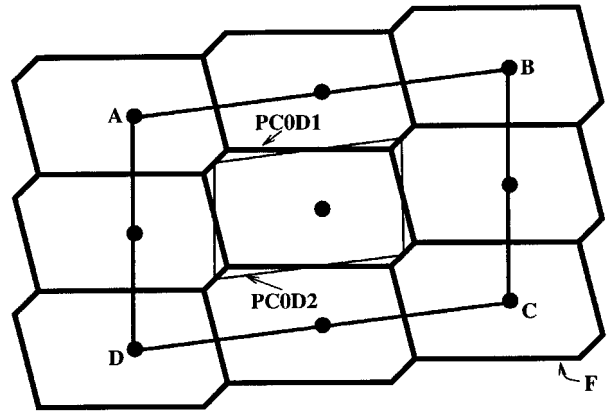

FIGURE 13. Every point in PCDKLM is also in F.

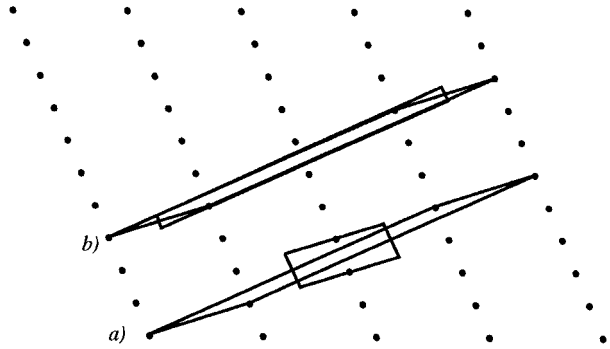

FIGURE 14. (a) When the vectors K, L, M are not ordered such that $|\boldsymbol{K}| \geq |\boldsymbol{L}| \geq |\boldsymbol{M}|$, the boxes PCDKLM and PCDUVW possibly have little overlap, so it is also possible that particles require shifts over high order lattice vectors to be moved from one box into another. (b) The boxes PCDKLM and PCDUVW have a large overlap because the longest box vector is called $\boldsymbol{K}$.

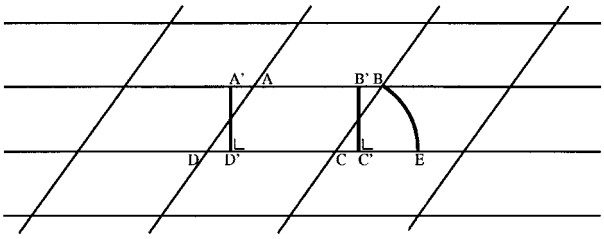

FIGURE 15. When the vectors $\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{M}$ are ordered such that $|\boldsymbol{K}| \geq |\boldsymbol{L}| \geq |\boldsymbol{M}|$, every point belong to PCDUVW is also in F.

Now, just like in Appendix A, we will determine the maximum shift required to bring a particle from PCDKLM into PCDUVW. We suppose that the vectors $\boldsymbol{K}$ and $\boldsymbol{L}$ are ordered; that is, $|\boldsymbol{K}| \geq |\boldsymbol{L}|$. We define ''F'' as the array of nine cells, created by all possible first order shifts of PCDKLM (Fig. 15). To show that every point of PCDUVW is in F it is sufficient to show that C′ is in F. This last statement can be reformulated as: show that the distance of C′ to C is less than the distance C to D. This last statement is true because $CC' < CE \leq DC$. So, at most, first order shifts are required to bring particles from PC0DKLM into PC0DUVW.

For the 3D case, the above reasoning can be applied twice. Thus, in 3D, at most, second order shifts are required to bring particles from PCDKLM into PCDUVW.

References

1. Fejes Tóth, *Regular Figures*, Pergamon Press, London, 1964, pp. 114-119.
2. W. F. van Gunsteren and H. J. C. Berendsen, *Groningen Molecular Simulation (GROMOS) Library Manual, Biomos*, Groningen, The Netherlands, 1987.
3. H. Bekker, E. J. Dijkstra, H. J. C. Berendsen, and M. K. R. Renardus, *Mol. Sim.*, **14**, 137 (1995).
4. H. Minkowski, *Allgemeine Lehrsätze über die konvexen Polyeder, Nachr. Ges. Wiss. Göttingen, Math.-Phys. Kl.*, or in the collected works *Gesammelte Abhandlungen*, Chelsea Press, New York, 1967.
5. W. Dzwinel, J. Kitowski, and J. Mościński, *Mol. Sim.*, **7**, 171 (1991).
6. *Handbook of Convex Geometry, Vol. B*, P. M. Gruber and J. M. Wills, Eds., North-Holland, Amsterdam, 1993.
7. H. Bekker, H. J. C. Berendsen, E. J. Dijkstra, S. Achterop, R. van Drunen, D. van der Spoel, A. Sijbers, H. Keegstra, B. Reitsma, and M. K. R. Renardus, in *Conference Proceedings Physics Computing '92*, World Scientific Publishing Co. Singapore, 1992, p. 257.
8. D. J. Adams, *Chem. Phys. Lett.*, **62**, 329 (1979).
9. S. S. Wang and J. A. Krumhansl, *J. Chem. Phys.*, **56**, 4287 (1972).
10. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, *J. Chem. Phys.*, **81**, 3684 (1984).